Lab 3: ADC and UART

Introduction

For lab 3, you'll be using everything you've learned so far in order to use the TM4C123's Analog to Digital Converters (ADC) to measure readings from a temperature sensor. Then, you'll use the board's Universal Asynchronous Receiver/Transmitter (UART) hardware transmit the temperature data serially.

By the end of this lab, you will:

- Have even more experience implementing timers and interrupts.
- Be familiar with the TM4C123's analog to digital converters.
- Know how to use analog sensors in a digital system.
- Understand the relationship between clock speed and power dissipation.
- Have experience configuring and using a UART interface.

Task 1: Analog to Digital Temperature Readings

The ADC Module

The ADC module of the TM4C123GH6PM features a 12-bit resolution and supports 12 input channels plus an internal temperature sensor. The ADC module uses a Successive Approximation Register (SAR) architecture to deliver a 12-bit, low-power, high-precision conversion value. The range of this conversion value is from 0x000 to 0xFFF (0 to 4095). It uses internal signals VREFP and VREFN as references to produce a conversion value from the selected analog input. VREFP is connected to VDDA and VREFN is connected to GNDA, as shown in Figure 1.

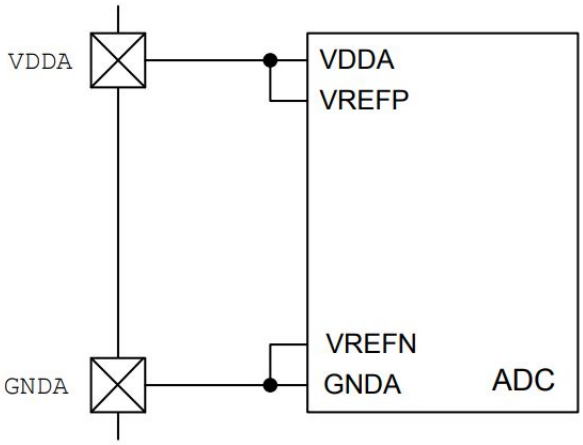

Figure 1 ADC Voltage (Figure 13-8 in the datasheet)

More detailed information about the ADC module on the TM4C can be found on page 1389 of the datasheet. To learn how to use the digital readings of the ADC, check section 13.3.4 of the datasheet.

To configure the ADC, follow the steps from section 13.4 and/or the notes from lecture. Some things to note:

- After configuring the **RCGCADC**, you might need to insert a small delay before moving on
- You will be sampling the ADC by triggering it with a timer. As such, you'll have to configure the ADC to be triggered by your timer *and* configure the timer to enable it to trigger the ADC.
- While the ADC is triggered by the timer, the timer does not need an interrupt attached to it. Instead, you need to enable the interrupt corresponding to the ADC. This means you'll be going through a lot of similar steps as you did in lab 2, such as updating the vector table.
- Since only one sample is needed, Sample Sequencer 3 is a good choice. You can read the ADC value from the **ADCSSFIFO3** register.

Internal Temperature Sensor

The built-in internal temperature sensor of the TM4C LaunchPad notifies the system when the internal temperature is too high or too low for reliable operation. It converts temperature into voltage according to the plot and equation shown in Figure 2.

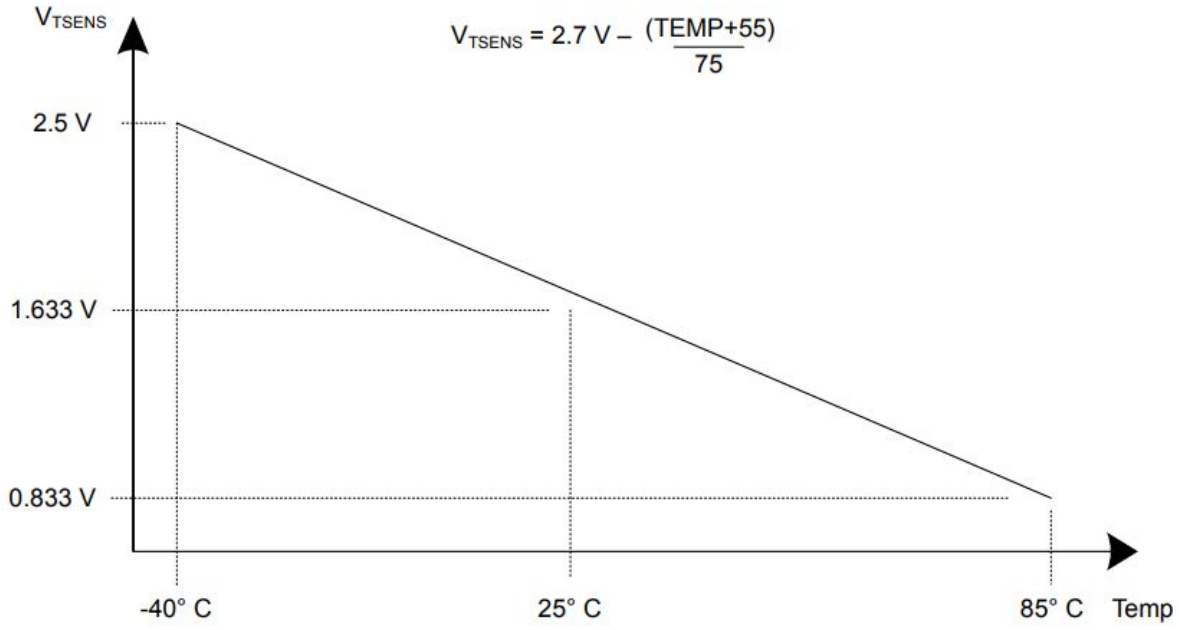

Figure 2 Internal Temperature Sensor Characteristic (Figure 13-11 in the datasheet)

This sensor voltage is read by the ADC module. Check out section 13.3.6 to get more information about how to convert the ADC reading into a temperature value.

Task 1 Assignment

Your task is to program the user switches on the TM4C to control the clock speed on the board while monitoring the internal temperature:

- When PF0 is pressed, change the system clock to begin operating at 4 MHz.
- When PF4 is pressed, change the system clock to begin operating at 80 MHz.
- A timer should trigger the ADC to interrupt the code once every second to read the current internal temperature on the board
- Calculate the temperature in Celsius and turn on the user LED based on the following chart:

Color	{PF3, PF2, PF1}	Temperature, °C
Red	0b001	0-17
Blue	0b010	17-19
Violet	0b011	19-21
Green	0b100	21-23
Yellow	0b101	23-25
Light Blue	0b110	25-27
White	0b111	27-40

You should find that setting the system to a lower clock speed, the light will tend towards the colors at the top of the table. Then, hitting PF4 should cause the temperature to rise, and the color will shift towards the colors at the bottom of the table.

Your program should have the following functions:

1. **Timer0_Init()**: This function will initialize the timer, per the instructions in lab 2. Configure the timer to trigger the ADC. Keep in mind that the clock speed won't be constant this lab, so you'll want a way to pass the maximum counter value as a variable.
2. **PortF_Init()**: This function will configure the GPIO Port F pins, as you've done before. The switches and all 3 user LEDs are being used here.
3. **PLL_Init()**: This function will initialize the Phase Locked Loop. Check page 21 of the lecture 5 slides, or section 5.3 of the datasheet.
4. **ADC_Init()**: This function will configure the ADC module, as described above. Section 13.4 of the datasheet will be helpful here.
5. **ADC0_Handler()**: This is the ISR for the ADC0 module. Here, you will read the ADC conversion value and calculate the temperature of the internal temperature sensor. Make sure to clear the ADC interrupt flag after each conversion.
6. **main()**: Here, you should call the initialization functions, and poll the user switches.

Once again, note that you're required to use your own header files. Do not include tm4c123gh6pm.h.

Task 2: UART

Universal Asynchronous Receiver/Transmitter

The Universal Asynchronous Receiver/Transmitter (UART) is a commonly used hardware interface for simple asynchronous serial communication. In this part of the lab, you will be using the TM4C's UART hardware to serially communicate with the board from your computer.

First, you will need a program to setup serial communication. A common option that can be installed on Windows, macOS, and Linux is PuTTY. Install PuTTY, or another program of your choice. The instructions henceforth will assume you are using PuTTY.

You'll need to use PuTTY to setup a communication link with the TM4C board. First, find the COM port associated with it. In windows, this can be found in the Device Manager. Here, right click on your board and hit properties, and then select Port Settings. Here, you will find the settings to configure in PuTTY.

In PuTTY, select "Serial," the bottom category on the left. Ensure the settings here match the settings from the Device Manager, and that you have selected the correct COM port. Then, select "Session," the top category on the left. Select "Serial" as your connection type.

You're now ready to begin serial communication with the board! Or, at least, your computer is. You still need to configure the UART hardware. To start, check out section 14.4 on page 902. PA[1:0], corresponding to UART0, is what you need.

Task 2 Assignment

1. **Use UART serial communication to print the temperature readings from Task 1 to the terminal in PuTTY.**

That's it! With the datasheet and all of the skills you've gained this quarter, you have everything you need to get serial communication working!

If you have extra time and want a challenge (and maybe a badge), implement a way to send clock frequency to the board without needing to press the buttons.

Lab Demonstration

Demonstrate your design for the assigned tasks to the TA.

Submission

Submit a short lab report that includes the following sections:

1. Procedure
    - Describe how you approached the assigned tasks and include any relevant diagrams, schematics, etc.
2. Results
    - Describe the results obtained for the assigned tasks. Give a brief overview of the finished product, as it relates to what was asked of you.
    - Include any relevant images, screenshots, etc.
3. Problems Encountered and Feedback

    - Describe any issues you had while completing the lab, and how/if you were able to overcome them.
    - Include any feedback you have about the lab, or any tips and tricks that you learned that may be useful to you in future labs.
    - Include how many hours you spent in completing this lab.
4. Appendix
    - In your Appendix, include any code you wrote in the lab. This should be **formatted** and **commented.**

   A good resource for formatting code for Word documents is found here: http://www.planetb.ca/syntax-highlight-word

Additionally, submit your commented .c and .h files.

Grading

Your grade for this assignment will depend on the following:

- Lab demonstration: **50%**
    - Your lab demonstration will be graded based on your completion of the assigned tasks, your ability to answer questions, and your understanding of the course material.
- Lab report: **20%**
    - Your lab report should be short and concise.
- Code comments: **30%**
    - Comment your code! Your code should have function comments and in-line comments where appropriate.